Basic Electronic Design for Proposed NMSU Hitchhiker Payload

by

Stephen Horan

NMSU-ECE-00-011

NMSU-ECE-00-011

Basic Electronic Design for Proposed NMSU Hitchhiker Payload

Stephen Horan

Manuel Lujan Space Tele-Engineering Program
New Mexico State University
Las Cruces, NM

Prepared for

NASA Goddard Space Flight Center
Greenbelt, MD

under Grant NAG5-7520

September 7, 2000

Klipsch School of Electrical and Computer Engineering
New Mexico State University
Box 30001, MSC 3-O
Las Cruces, NM 88003-8001

Contents

List of Figures iv

List of Tables vi

Software Listings vii

INTRODUCTION . . . . -1-

DESIGN COMPONENTS . . . . -1-
- Safety Inhibits . . . . -1-
- Power . . . . -4-
- Payload Status Monitoring . . . . -6-
- Communications . . . . -13-
- Data Content, Form, and Format . . . . -14-
- Payload Data Processing . . . . -17-
- User Interface . . . . -25-

References . . . . -33-

List of Figures

Number	Title	Page
1	Inhibit switches and experiment power system enabling	2
2	Illustration of the Crew Switch and Lid Switch indicators on the User Interface	3
3	Illustration of experiment supply voltage monitoring on the User Interface	3
4	Computer and monitoring power supply in PC-104 standard format	5
5	Pressure and temperature sensor array	7
6	Power relay distribution circuit diagram	8
7	Power relay distribution circuit prototype	9
8	Voltage regulator circuit diagram	10
9	DC-to-DC converter, voltage regulators, and inhibit switches	11
10	PC-104 relay board	12
11	Pressure and temperature sensors	12
12	PC-104 33 MHz, 386 computer with serial ports for command and telemetry	14
13	Command word and telemetry frame formats	15
14	PC-104 modules forming the processing hardware	18
15	Software and processing hardware development environment	18
16	Analog-to-digital conversion board with 3 digital output ports in PC-104 standard board format	19
17	VGA video display board in a standard PC-104 format used for development and testing	20
18	Total user interface display	27
19	User interface program diagram	28
20	Command portion of the user interface	29
21	Command processing portion of the overall LabVIEW user interface VI	29

List of Figures

Number	Title	Page
22	Level 0 processing LabVIEW front panel	31
23	Level 0 processing LabVIEW functional diagram	31
24	Level 1 processing LabVIEW front panel	32
25	Level 1 processing LabVIEW functional diagram	32

List of Tables

Number	Title	Page
1	Valid command table	15
2	Telemetry table	16

Software Listings

Number	Title	Page
1	Payload software for command and telemetry processing	21

INTRODUCTION

This document presents the basic hardware design developed by the EE 499 class during the spring semester of the 1999-2000 academic year. This design covers the electrical components to supply power to the experiments, the computer software and interfaces to control the experiments, and the ground data processing to provide an operator interface. This document is a follow-on to the Payload Mission description document [1] and the System Requirements document [2] developed during the EE 498 class during the fall semester.

The design activities are broken down by functional area within the structure. For each area, we give the requirements that need to be met and the design to meet the requirements. For each of these areas, a prototype selection of hardware and/or software was done by the class and the components assembled as part of the class to verify that they worked as intended.

DESIGN COMPONENTS

The basic payload infrastructure design components are described here. Each component is identified by major function. The experimenter payload designs are not included since they are still under development. Background information for the design can be found in the CPR baseline documentation [3] and the CARS documentation supplied by NASA [4].

1. Safety Inhibits
   a. Requirements
      i. Provide for three inhibits to prevent power from being applied to the experiments until and unless permitted by payload operations.
      ii. Provide for the associated state monitoring of the inhibits and the telemetering of the state information to the payload operator.
   b. Design
      i. Inhibit Mechanism

      The inhibit mechanism consists of the following parts:

      (1) Crew Switch
      (2) Lid Switch
      (3) Power distribution relays

      The purpose of the Crew Switch and Lid Switch are to provide independent enable signals to the DC-DC converter that keeps power from the experiments unless both the crew has activated the switch and the lid to the Hitchhiker canister has opened. If either switch is open, the transistor will not supply sufficient voltage to the On/Off input of the DC-DC converter. When the DC-DC converter is enabled, it supplies the input to the voltage regulators which power the experiments under control of the power distribution relays. The circuit diagram for the switch inhibit mechanism is illustrated in Figure 1. The third independent inhibit is provided via the on-board computer enabling the power distribution

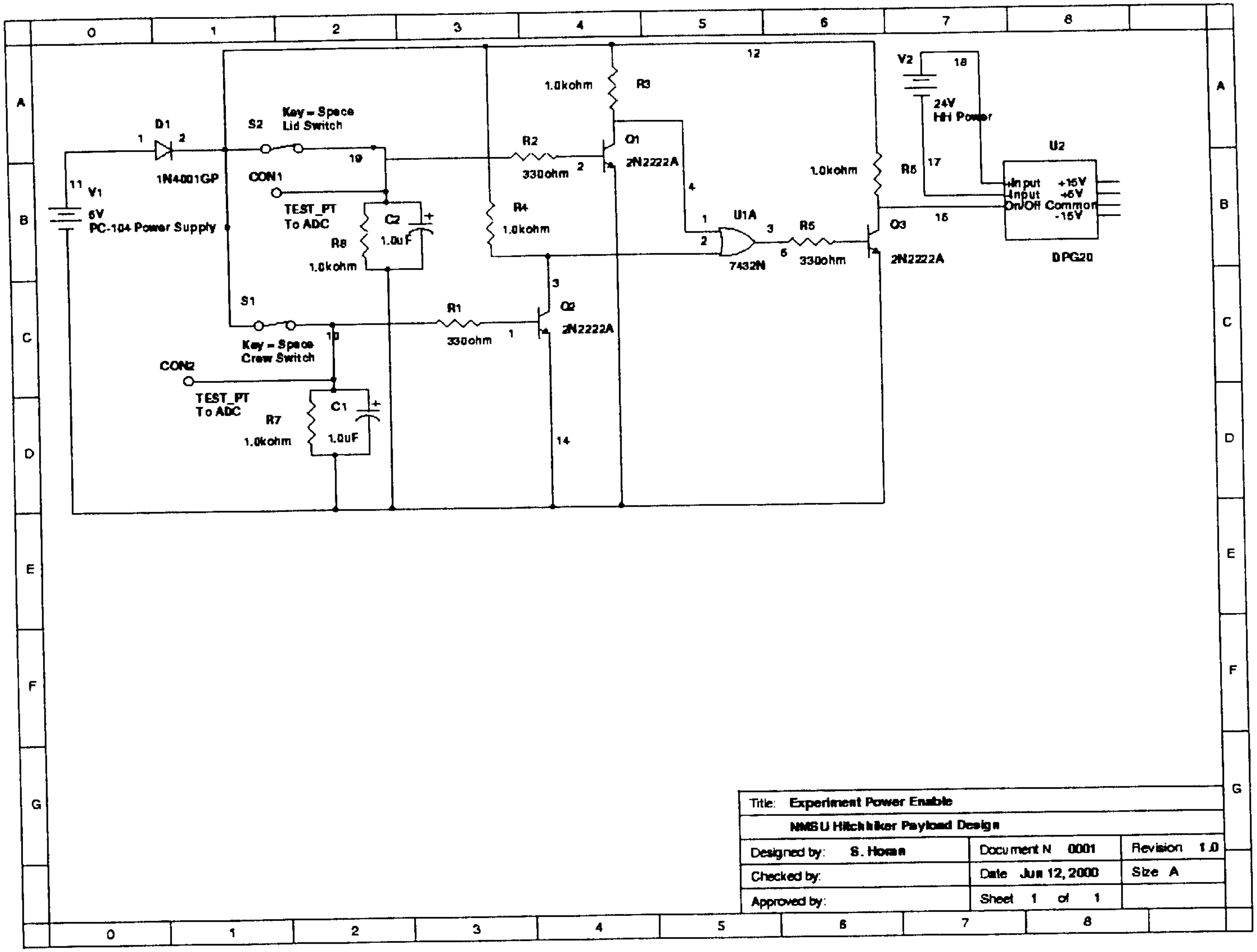
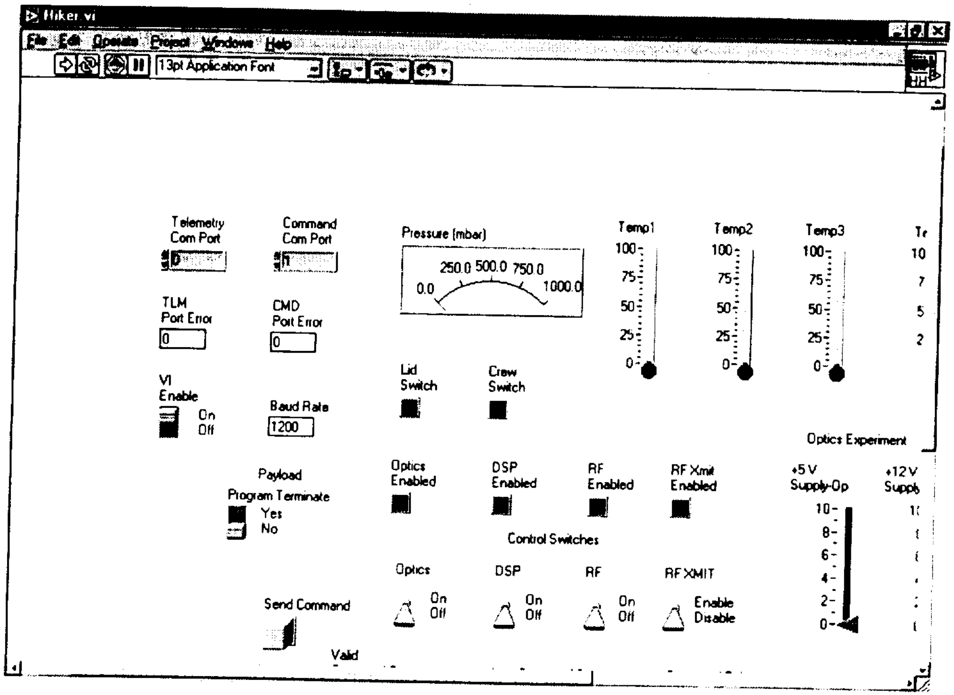


Figure 1 - Inhibit switches and experiment power system enabling.

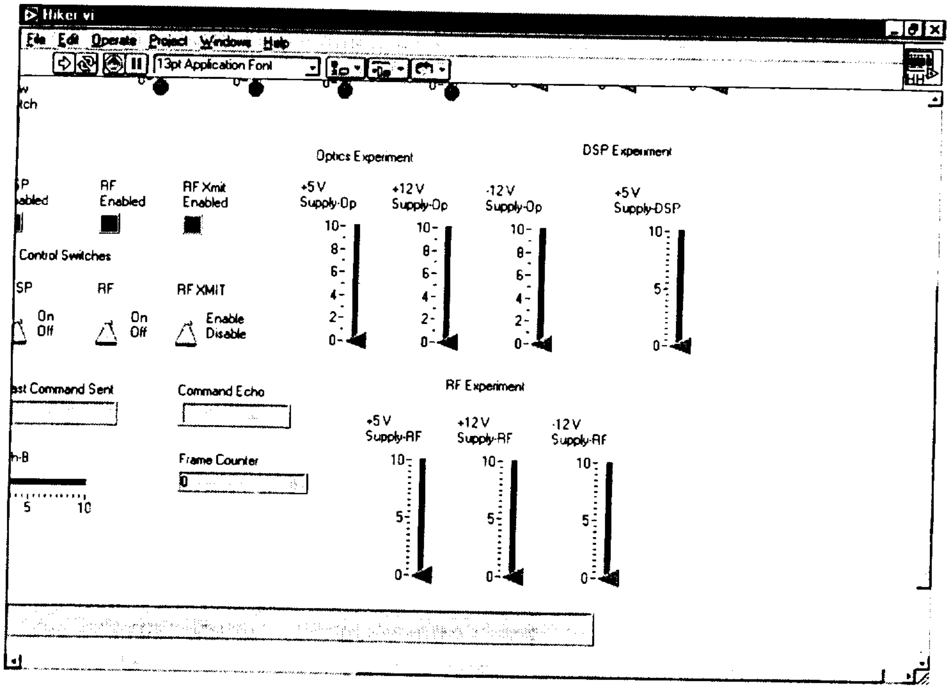


TLM Port Error 0

CMD Port Error 0

Baud Rate 1200

Figure 2 - Illustration of the Crew Switch and Lid Switch indicators on the User Interface.



Frame Counter 0

Figure 3 - Illustration of experiment supply voltage monitoring on the User Interface.

relays. The power distribution relays are controlled by the ground operator. The relays are normally OFF so that no power is given to the individual experiments. Only when the operator explicitly commands the relay for the individual experiment to be energized will the experiment be permitted to receive power.

ii. Inhibit State Monitoring

The monitoring system telemeters the state of the Crew Switch and the Lid Switch, as measured at points CON1 and CON2 in Figure 1, to the ground so that the operators can verify correct switch state prior to commanding the payload. This is shown in Figure 2 where a portion of the User Interface is illustrated. The two lights labeled "Crew Switch" and "Lid Switch" will show the current state (green = enabled; red = disabled) to the payload operator.

The operators can also verify that power is not connected to the experiments by looking at the voltage level indicators for each experiment on the User Interface. These meters are shown in Figure 3.

2. Power

a. Requirements

i. Provide for power to allow monitoring of the inhibit state whenever the payload is active.

ii. Provide for power to allow monitoring of temperature and pressure sensors whenever the payload is active.

iii. Provide the means to enable and disable power to each experiment individually by operator command.

iv. Provide the means to monitor voltage levels for the power supplied to each experiment.

v. Provide voltages at + 12V, + 5V, and -12 V for use in each experiment.

vi. Operate from the power supply provided voltage specified in HH CARS.

vii. Current draw and total power not exceed limits provided in HH CARS.

b. Design

i. Monitoring Power

To provide power to monitor the inhibit states and provide power for the on-board payload processing, a dedicated power supply was chosen. This power supply is illustrated in Figure 1 where it supplies power to the inhibit circuitry for the DC-DC converter. It is a standard PC-104 format power supply as shown in Figure 4 that will provide necessary power for the computer system and the analog data acquisition system. It is active whenever standard power is applied to the payload.

The power for the temperature and pressure sensors is provided by the data acquisition system directly. This is illustrated in Figure 5. The data acquisition system is powered via the PC-104 bus using the power supply from Figure 4 as

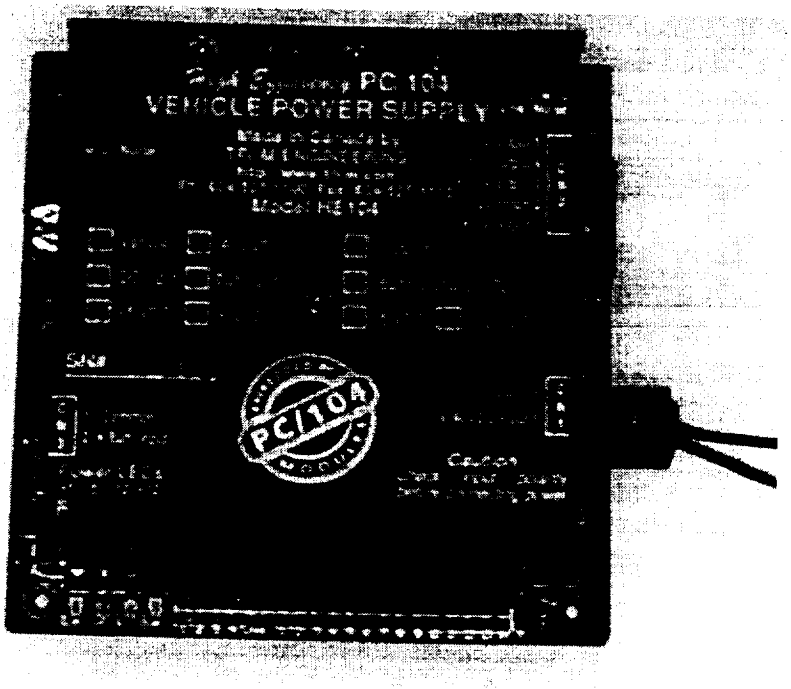


Figure 4 - Computer and monitoring power supply in PC-104 standard format.

the source.

ii. Controlled Experiment Power Distribution

The power distribution control is accomplished by two independent circuits: a DC-DC converter and an experiment power relay system. As illustrated in Figure 1, the DC-DC converter is equipped with an enable to prevent output electricity distribution unless the enable is activated. The enable is activated by the inhibit switches mentioned above. The relay system is controlled by the payload computer based upon explicit operator command from the ground. If the DC-DC converter is enabled, the converter produces a +15 V, a +5 V and a -15 V output. The +15 V output is used to drive +12 V regulators to supply regulated voltage to the experiments while the -15V output is used to drive -12 V regulators to supply that voltage to the experiments needing it. The relay system then routes the +12 V, +5 V, and -12 V to the experiments needing each voltage. The voltages have a unique relay for each experiment so that only the active experiments receive any power. The relay network circuit diagram is illustrated in Figure 6. The prototype circuit is shown in Figure 7.

iii. Power Bus

The power for the experiments originates with the basic shuttle power. This is used by the DC-DC converters to provide electricity to two voltage regulators for + 12 V and -12 V. The DC-DC regulator provides +5 V as one of its outputs The voltage regulator circuit diagram is illustrated in Figure 8. The prototype circuit containing the DC-DC converter, the enables, and the voltage regulators is illustrated in Figure 9. This is then the source of the power for the distribution relay circuit shown in Figures 6 and 7. As shown in Figure 8, there are three power plugs provided to supply the necessary voltages and ground for the experiments according to the following list:

(1) DSP experiment: +5 V
(2) RF experiment: +5 V, +12 V, -12 V
(3) Optics experiment: +5 V, +12 V, -12 V

Each of these power supplies is individually enabled via control signals sent to the relay circuit using the ribbon cable shown in Figure 7.

One improvement to this configuration is to use the PC-104 relay board illustrated in Figure 10. This board will allow the control computer to have more direct control of the relay board. This particular module also has optoisolated relay switching to improve noise immunity.

3. Payload Status Monitoring

a. Requirements

i. Provide for monitoring of the internal pressure of the payload

ii. Provide for the monitoring of payload temperatures at the four points as follows:

(1) optical experiment
(2) RF experiment
(3) CPU
(4) power supply

b. Design.

i. Pressure is sensed by a Motorola MPX5100A integrated pressure sensor as illustrated in Figure 11. Power is supplied by the ADC as was illustrated in Figure 5. The sensor voltage is measured by the ADC, also as was illustrated in Figure 5.

ii. Temperature is sensed at each location by a National Semiconductor LM35 temperature sensor as illustrated in Figure 11. Power is supplied by the ADC as was illustrated in Figure 5. The sensor voltage is measured by the ADC, also as was illustrated in Figure 5.

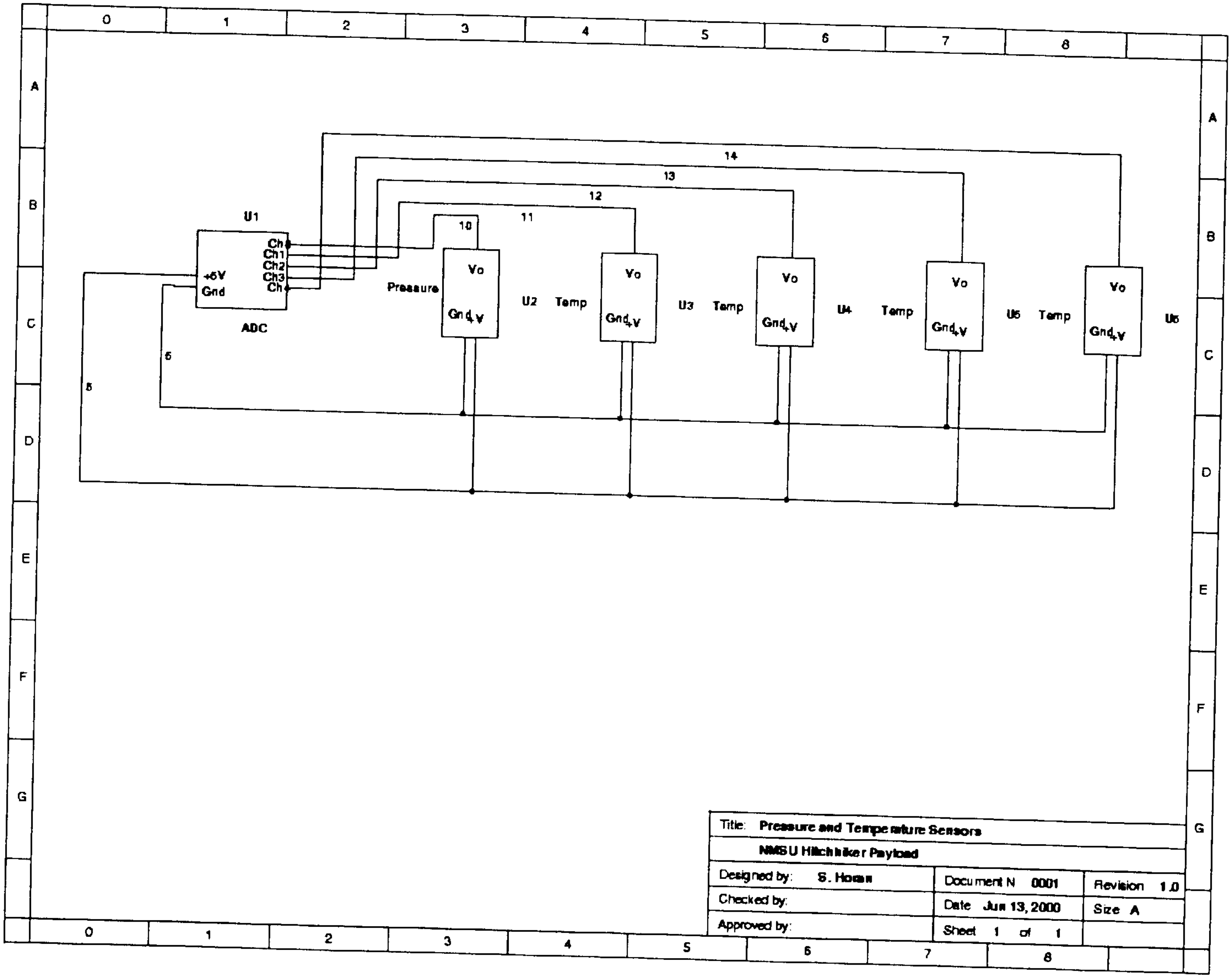


Figure 5 - Pressure and temperature sensor array.

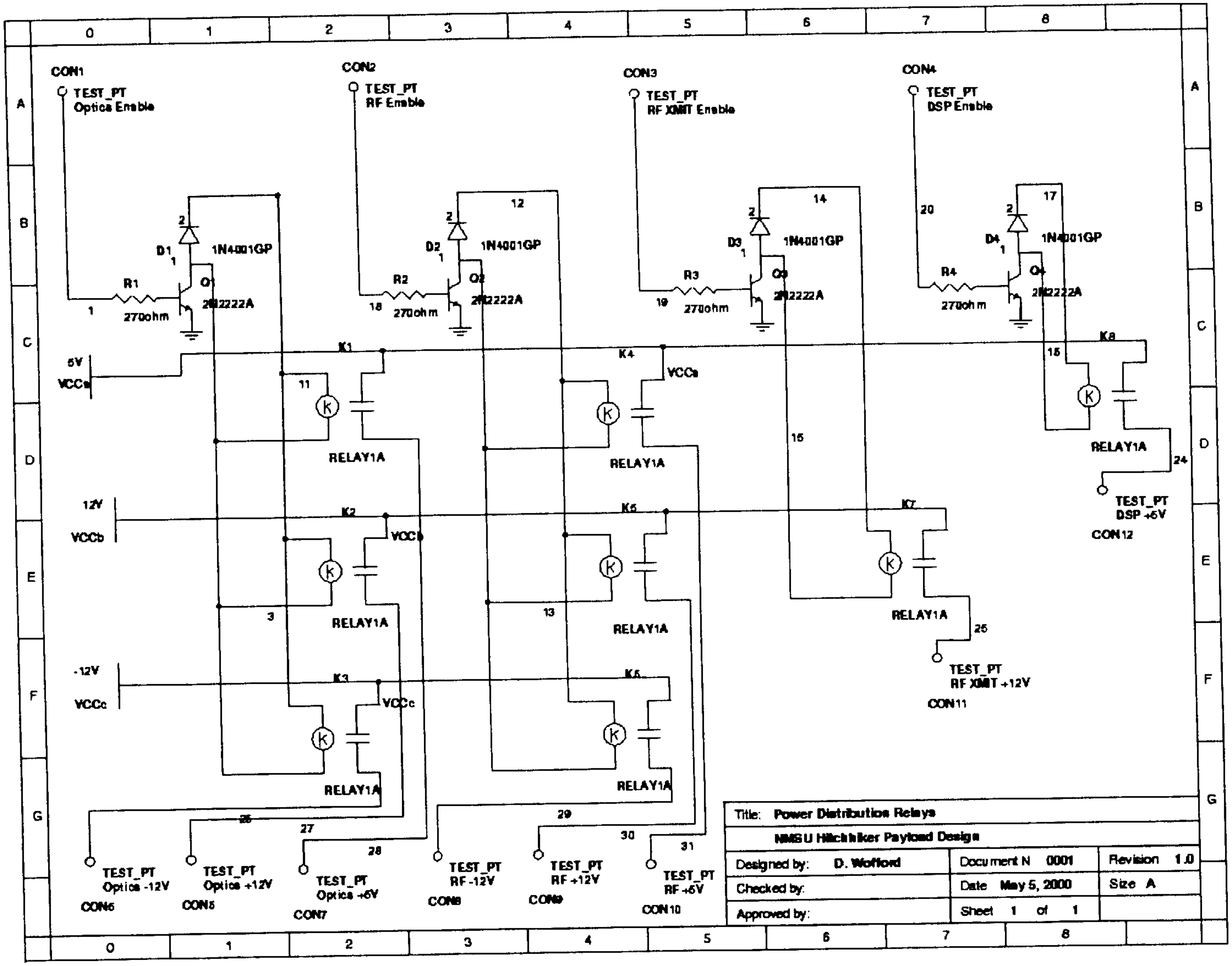


Figure 6 - Power relay distribution circuit diagram.

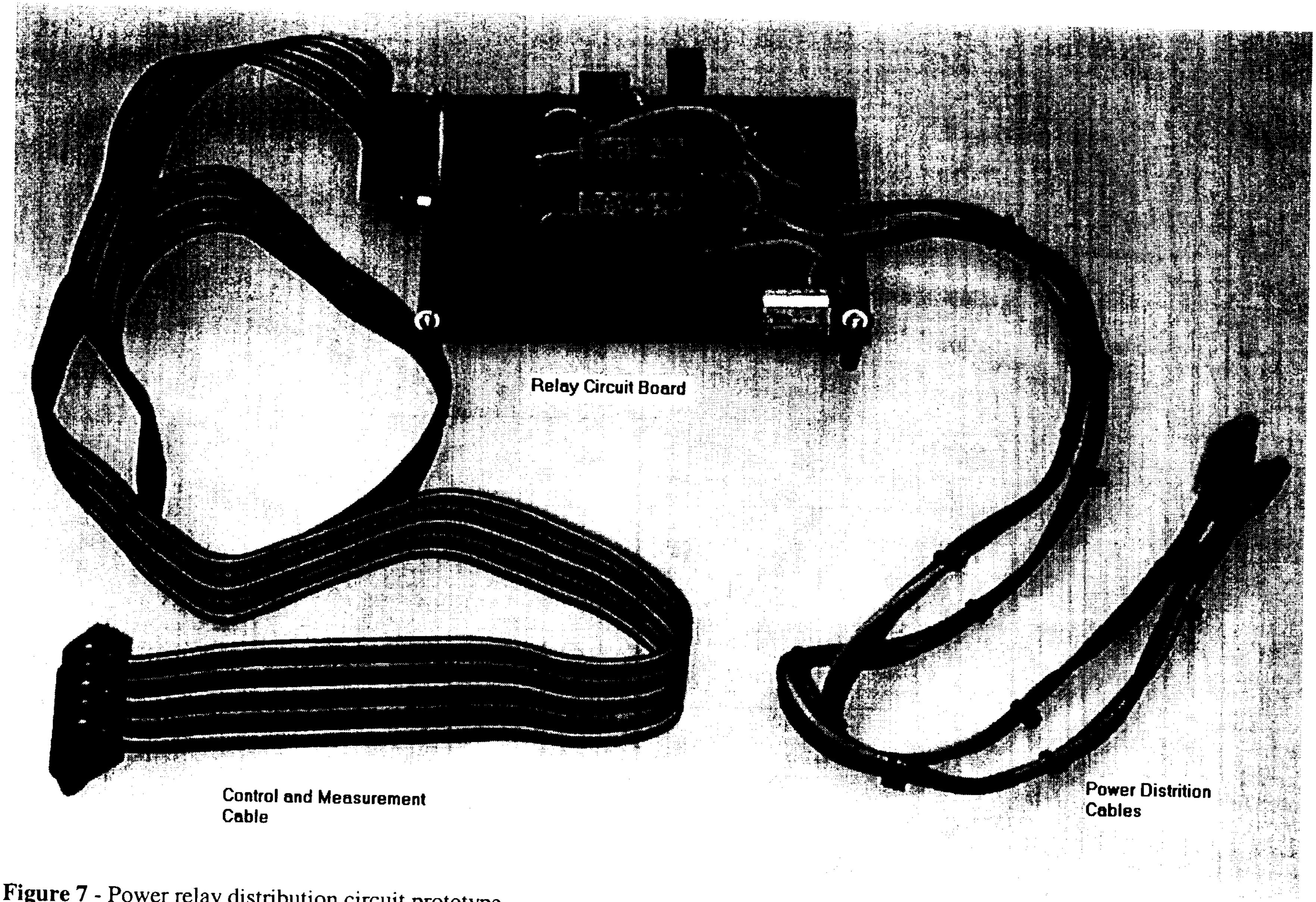


Figure 7 - Power relay distribution circuit prototype.

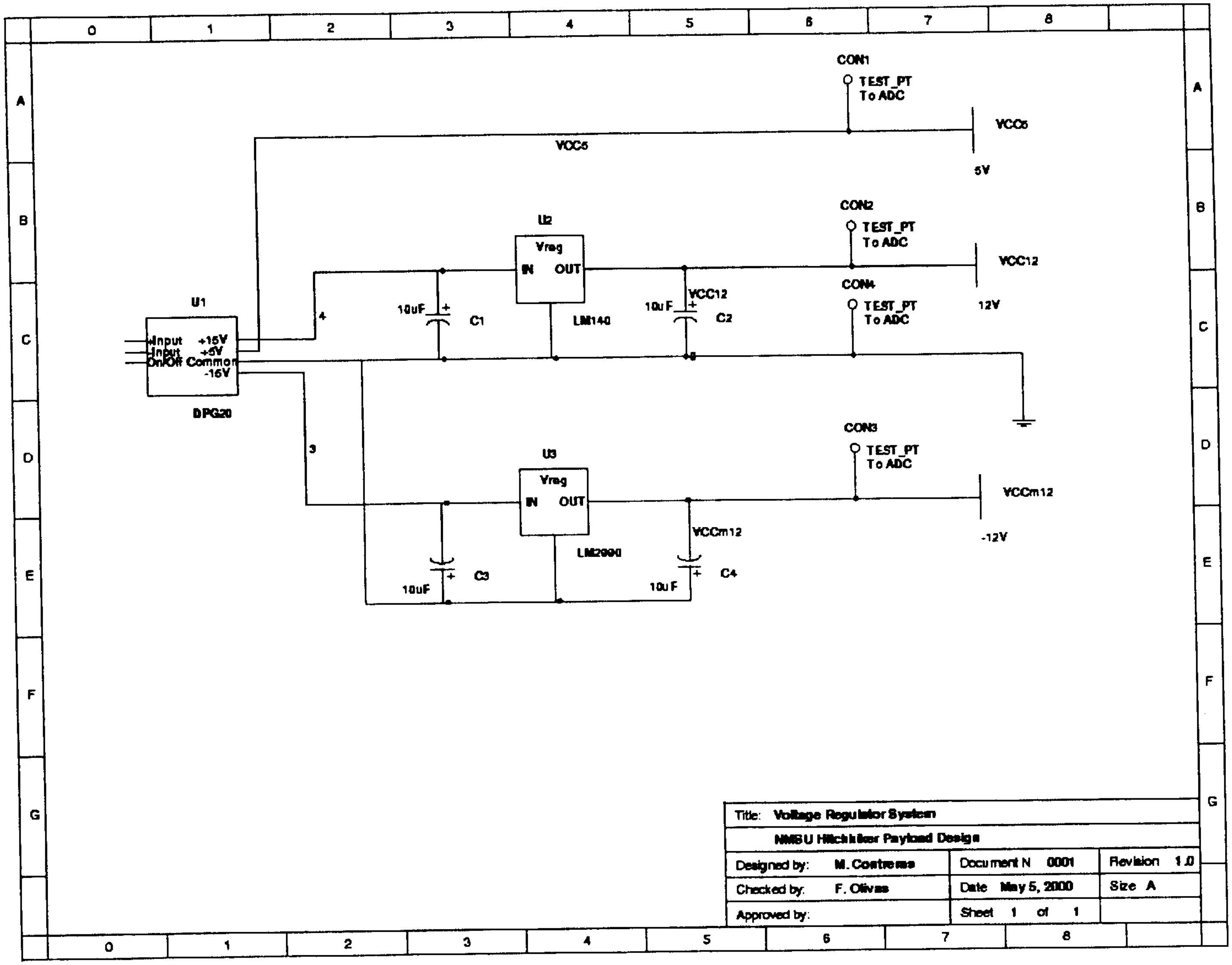


Figure 8 - Voltage regulator circuit diagram.

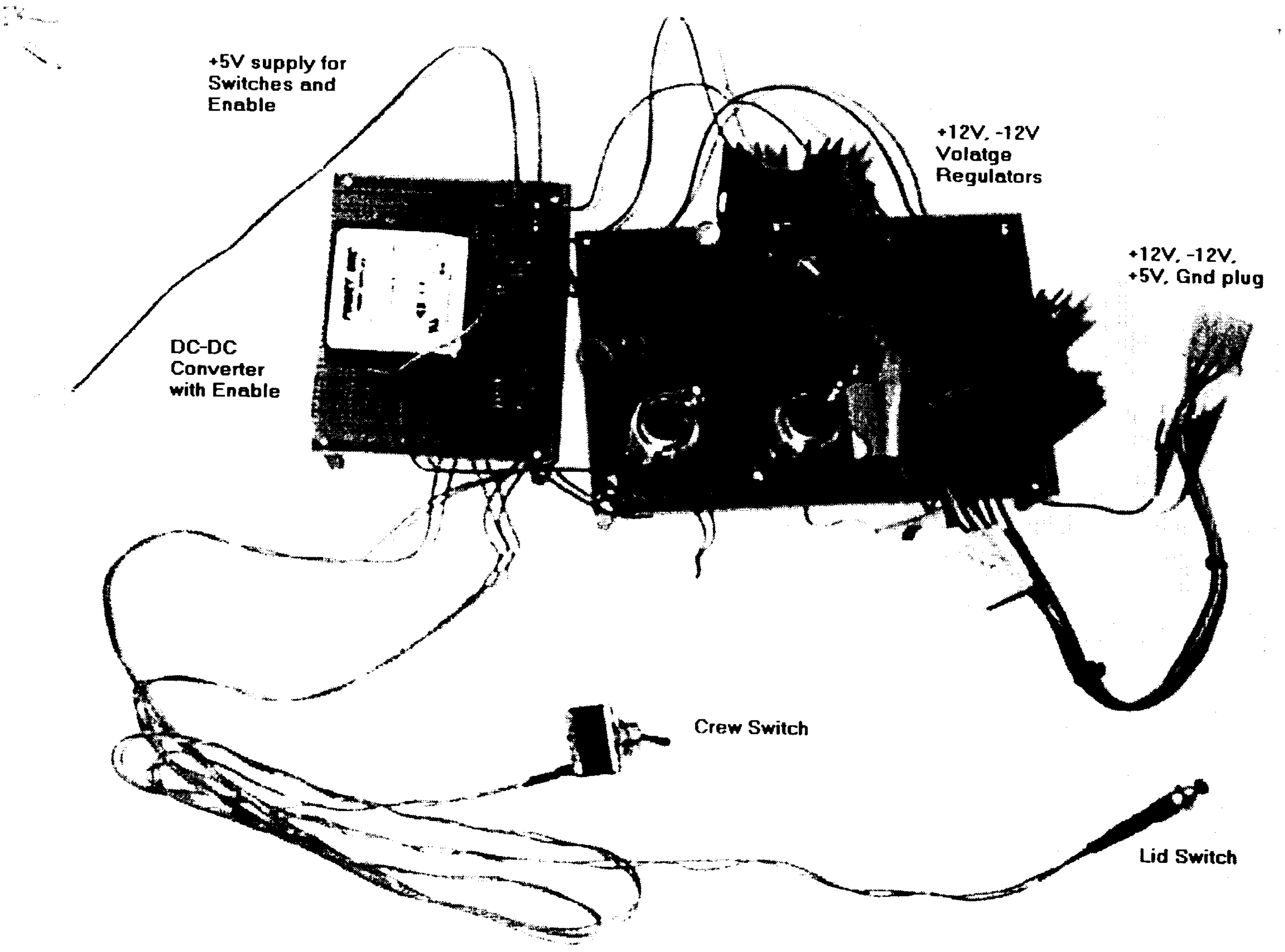


Figure 9 - DC-to-DC converter, voltage regulators, and inhibit switches.

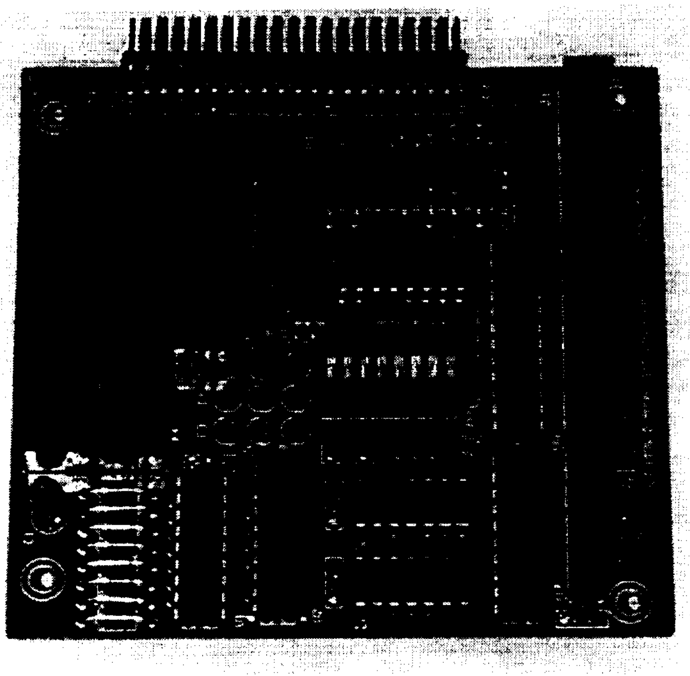


Figure 10 - PC-104 relay board.

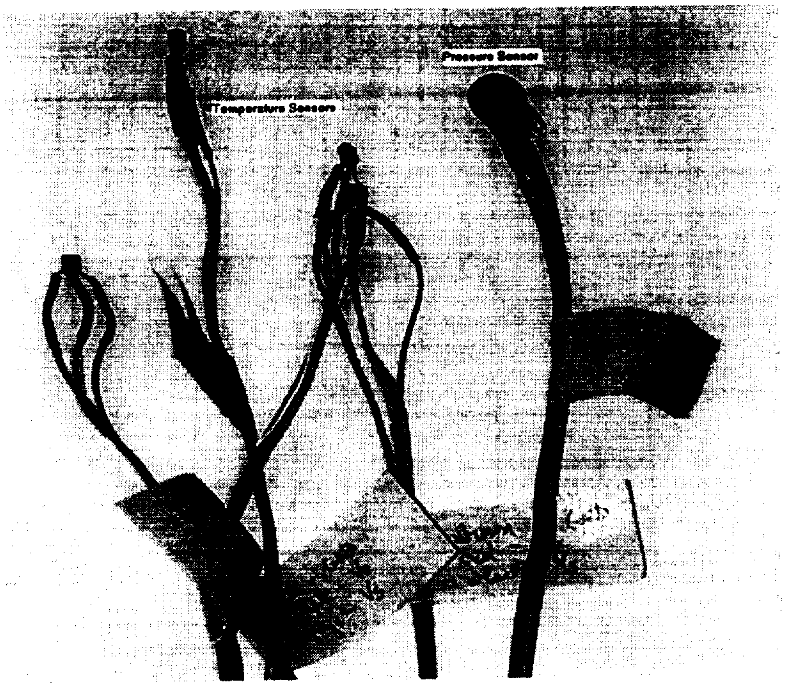


Figure 11 - Pressure and temperature sensors.

4. Communications

a. Requirements

i. Provide for command input using the HH forward asynchronous link at 1200 bps, 8 bits, no parity, 1 stop bit

ii. Provide for telemetry output using the HH return asynchronous link at 1200 bps, 8 bits, no parity, 1 stop bit

iii. Command interface to utilize cabling compatible with standard HH interface as in HH

iv. Telemetry interface to utilize cabling compatible with standard HH interface as in HH.

b. Design

i. Command Link

The command interface is designed using a standard RS-232 port on the payload computer. This port interfaces with the Hitchhiker command cable as described in HH. The computer's RS-232 port is configured as part of the initial software procedure in line #20 when the payload computer is activated. The command port is illustrated as part of the CPU illustration in Figure 12.

ii. Telemetry Link

The telemetry interface is designed using a standard RS-232 port on the payload computer. This port interfaces with the Hitchhiker telemetry cable as described in HH. The computer's RS-232 port is configured as part of the initial software procedure in line #21 when the payload computer is activated. The telemetry port is illustrated as part of the CPU illustration in Figure 12.

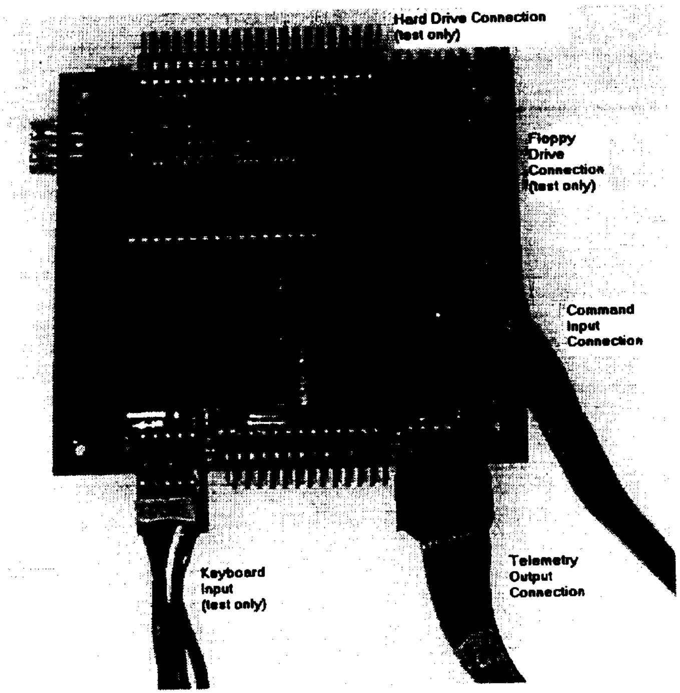


Figure 12 - PC-104 33 MHz, 386 computer with serial ports for command and telemetry links.

5. Data Content, Form, and Format
   a. Requirements
      i. All operator commands shall be formatted as described in the Valid Command Table (Table 1) prior to transmission to the payload. Each command shall begin with a synchronization word (146F hex) and formatted as shown in Figure 13.
      ii. All telemetry values shall be formatted into data frames as described in the Telemetry Table (Table 2) prior to transmission to the ground. Each telemetry frame shall begin with a synchronization word (EB90 hex) and formatted as shown in Figure 13.
      iii. All command and telemetry data are to be sent as ASCII representations of numbers or as text.

   b. Design
      i. Operator commands as described in the Command Table are formatted as one of the operations of the User Interface described later. Each command is validated upon reception at the payload by the control program given in Listing 1.
      ii. Telemetry values as described in the Telemetry Table are formatted in the payload control program given in Listing 1. The User Interface described below interprets the values and presents the data to the payload operator.

Table 1 - Valid Command Table		
Command Number	**Command Value**	**Meaning**
1	00	Disable all experiments
2	01	Optics Experiment Enable
3	02	DSP Experiment Enable
4	04	RF Experiment Enable
5	08	RF Transmit Enable
6	16	Terminate payload control program

146F	Command Data

Command Word

Synch Word	Temper-ature 1	Temper-ature 2	Temper-ature 3	Temper-ature 4	-12 V Supply	Optics +5 V	Optics +12 V	RF -12 V	RF +5 V	RF XMIT +12 V

DSP +5 V	+12 V Supply	+5 V Supply	Optics -12 V	RF XMIT -12 V	Lid Switch	Crew Switch	Frame Count	Command Count	Payload State	Command Echo

Telemetry Frame

Figure 13 - Command Word and Telemetry Frame formats.

Table 2 - Telemetry Table

Parameter Number	Type	Format	Meaning
1	Digital	Hex4	Synch Word (EB90)
2	Analog	Hex4	Pressure
3	Analog	Hex4	Temperature 1
4	Analog	Hex4	Temperature 2
5	Analog	Hex4	Temperature 3
6	Analog	Hex4	Temperature 4
7	Analog	Hex4	-12 V Supply
8	Analog	Hex4	Optics +5 V
9	Analog	Hex4	Optics +12 V
10	Analog	Hex4	RF -12 V
11	Analog	Hex4	RF +5 V
12	Analog	Hex4	RFXMIT +12 V
13	Analog	Hex4	DSP +5 V
14	Analog	Hex4	+12 V Supply
15	Analog	Hex 4	+5 V Supply
16	Analog	Hex4	Optics -12 V
17	Analog	Hex4	RF -12 V
18	Analog	Hex4	Lid Switch State
19	Analog	Hex4	Crew Switch State
20	Digital	Hex4	Frame Count
21	Digital	Hex4	Command Count
22	Digital	Hex4	Payload State
23	Digital	I2	Command Echo (146F,I2)

6. Payload Data Processing

a. Requirements

i. All commands received in the payload shall be compared against the Valid Command Table prior to processing.

ii. Only valid commands shall be processed. In valid command shall be rejected.

iii. All commands received shall be echoed in telemetry regardless of validity.

iv. A running count of all valid commands received while the payload is active shall be maintained and sent as part of the telemetry data.

v. Telemetry frames shall be constructed and sent to the ground with all values for the Telemetry Table at most once per second.

vi. Telemetry frames shall be numbered with a consecutive counter and the count be part of the telemetry frame.

b. Design

i. Processing Hardware

The processing hardware was designed around components packaged in the industry-standard PC-104 format. This standard was chosen because the packaging is modular – each component plugs into the next one using a standard connector and spacers, each module performs a specific function and unnecessary functions can be eliminated, and the programming could be accomplished using standard DOS languages. In this particular case, Quick Basic was used since the total processing is rather small and there is no need for a user interface in the final flight system. The processing hardware was configured around a CPU, data acquisition system, and support components for development and testing. The processing hardware components are shown in Figure 14. The development environment for the software and processing hardware is illustrated in Figure 15.

(1) CPU

The CPU that was chosen was a 33-MHz, 386 processor. This particular comes packaged with DOS 6.22 in flash memory configured as a RAM disk drive. By choosing the jumpers correctly, the CPU can be configured for running with a hard disk drive and floppy disk drive. For development and testing, the CPU ran from the hard disk. Once completed, the control program can be compiled and saved to the RAM disk. The CPU is illustrated in Figure 12.

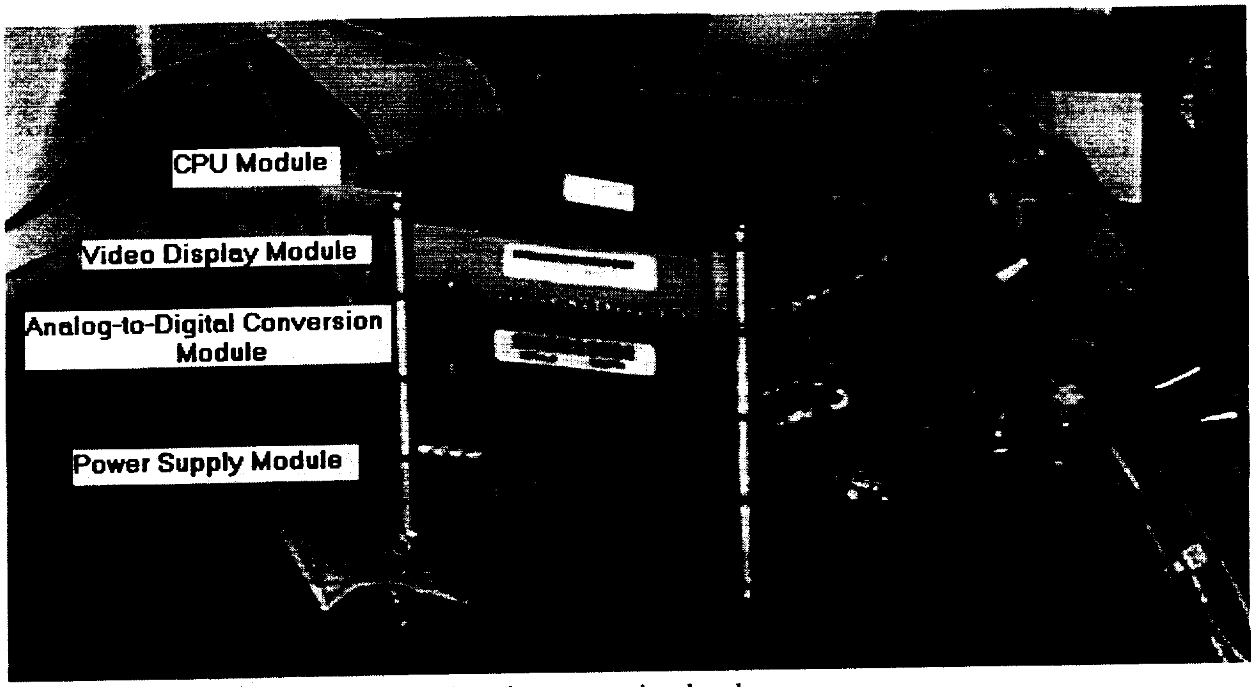


Figure 14 - PC-104 modules forming the processing hardware.

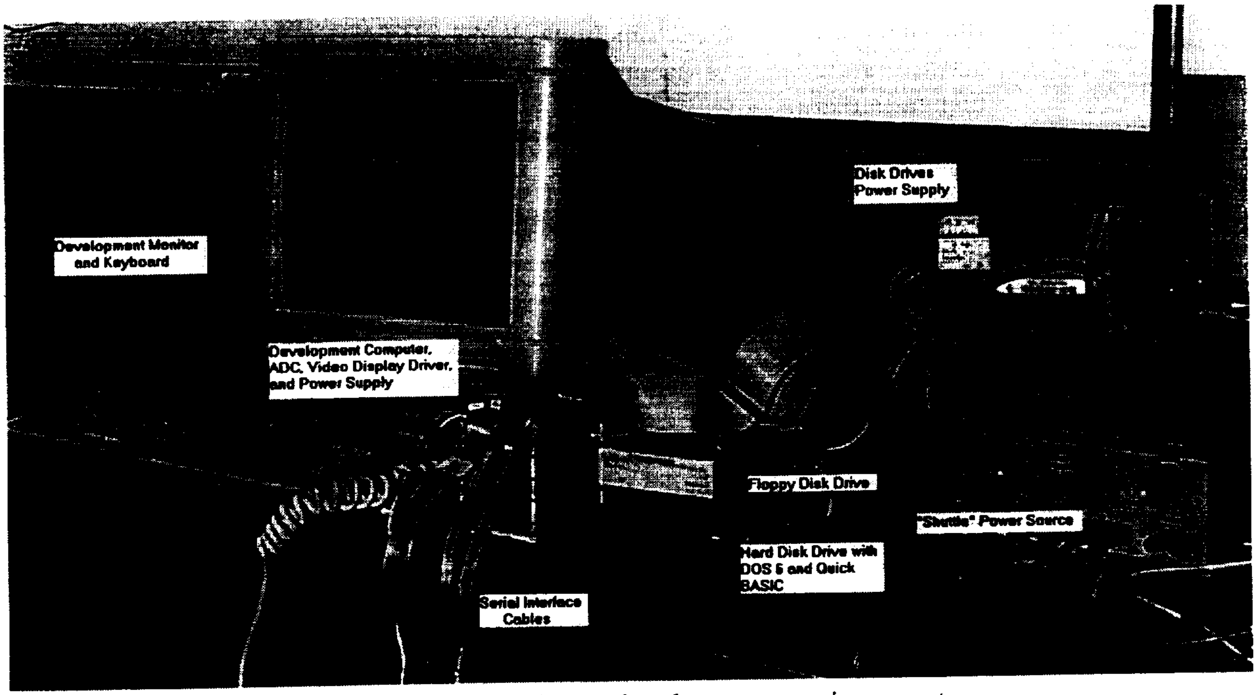

Disk Drives Power Supply
Development Monitor and Keyboard
Development Computer, ADC, Video Display Driver, and Power Supply
Floppy Disk Drive
"Shuttle" Power Source
Hard Disk Drive with DOS 5 and Quick BASIC
Serial Interface Cables

Figure 15 - Software and processing hardware development environment.

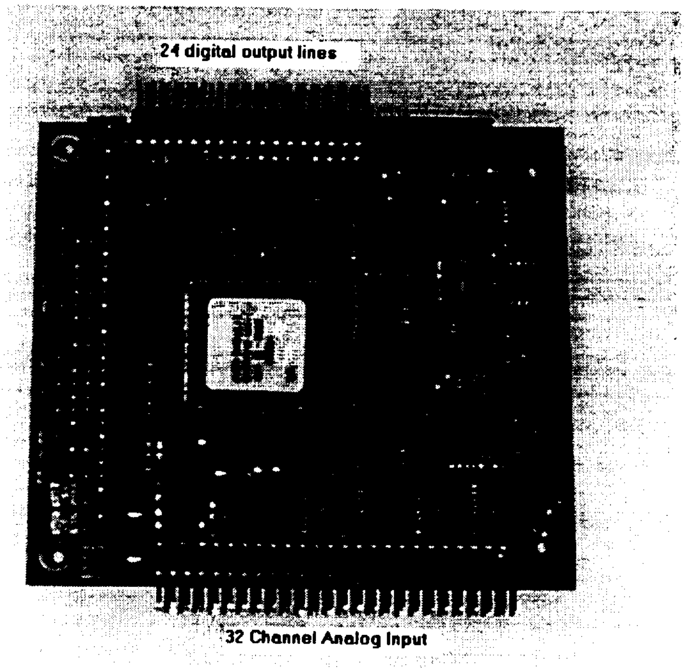


Figure 16 - Analog-to-Digital Conversion board with 3 digital output ports in PC-104 standard board format.

(2) Sensor Signal Acquisition

The sensor signals will be acquired with a data acquisition system provided in a PC-104 module. This particular module allows for 32 single-ended inputs or 16 differential inputs. It also has three digital output ports that are used to control the relay settings. The signal acquisition board is illustrated in Figure 16.

(3) Ancillary Connections for Test and Development

In order to make the system work for development and testing, several other capabilities were added to the PC-104 based CPU. These included a video display card, hard disk and floppy disk storage, and a keyboard.

(a) Video Card

The CPU did not come packaged with a VGA video driver so a separate video display unit was acquired for development and testing. This unit would normally be removed for flight. The card is in the standard PC-104 format so it plugs into the CPU board. The video display card is illustrated in Figure 17.

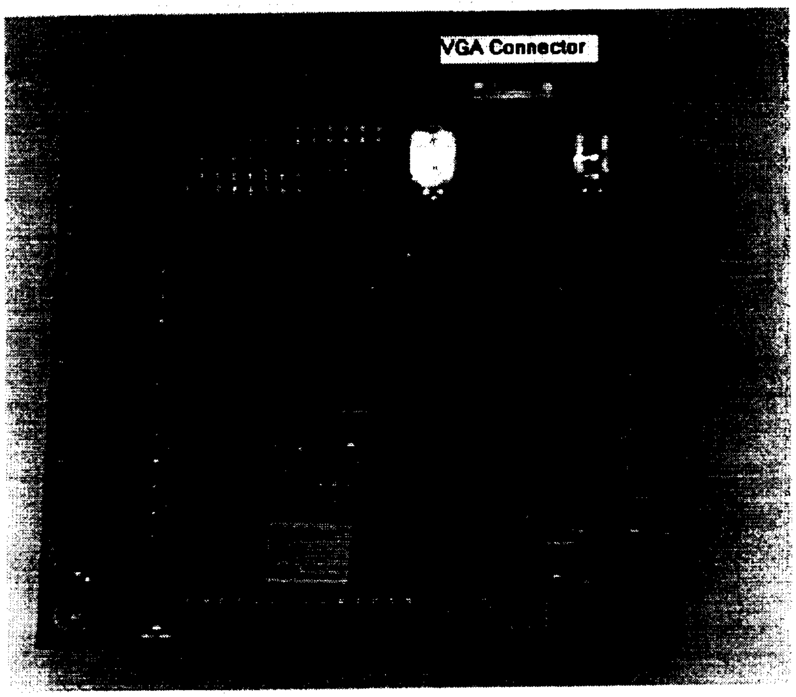


Figure 17 - VGA video display board in a standard PC-104 format used for development and testing.

(b) Storage

The CPU has an onboard RAM disk. For development and testing, a separate hard disk drive and floppy disk drive were added as shown in Figure 15. These plug into the CPU board using standard PC connectors. They would normally be removed for flight after the control software has been loaded onto the RAM disk.

(c) Keyboard

A standard PC keyboard is attached to the CPU module using the "utility" connector. This is for the development and test phase as shown in Figure 15 and would be removed for flight.

ii. Processing Software

The payload command and telemetry processing software has three major sections: payload configuration, telemetry processing, and command processing. The software listing follows in Listing 1. Each line is numbered and the numbering is used in the following discussion. Individual statements in the code are commented for better understanding of the functional details. The execution first performs the payload initialization. Then the software enters a large loop where the command and telemetry processing operates continuously until either the payload is powered down or the operators send a termination command to the

Listing 1 - Payload Software for Command and Telemetry Processing

```
CLS
OE% = 0
DSPE% = 0                                                'Optics Enable Flag
RFE% = 0                                                 'DSP Enable Flag
RFXE% = 0                                                'RF Enable Flag
PT% = 0                                                  'RF Transmit Enable Flag
EXPRUN% = &H0                                            'Program Terminate Flag
Cmd$ = ""                                                'Initialize experiment enable
CC% = 0                                                  'Initialize command buffer
FC% = 0                                                  'Initialize Valid Command Counter
CE$ = "146F,00"                                          'Initialize Frame Counter
Cfs$ = "146F,"                                           'Initialize Command Echo
'                                                        'Command Frame Synch string
'Set COM1 & COM2 ports for 1220 baud, N parity, 8 data bits, 1 stop bit
'ASCII data exchange, no hardware handshaking and 2K transmit & receive
'buffers
'COM1 = command input port
'COM2 = telemetry output port
'
OPEN "COM1:1200,N,8,1,ASC,CD0,CS0,DS0,OP0,RS,RB2048" FOR INPUT AS #2
OPEN "COM2:1200,N,8,1,ASC,CD0,CS0,DS0,OP0,RS,TB2048" FOR OUTPUT AS #1
'
'Set start and stop channel numbers
'
ch0% = 0
ch1% = 17                                                'start channel
fs$ = "EB90,"                                            'stop channel
adc% = &H300                                             'frame synch string
DO WHILE PT% <> 1                                        'base address of ADC board
frame$ = fs$                                             'run until terminate command
now& = TIMER                                             'initialize data frame
FOR K = ch0% TO ch1%                                     'get current time
OUT adc% + 2, K                                          'For each Data Channel
OUT adc% + 3, K                                          'set lower scan channel
OUT adc% + 11, 0                                         'set upper scan channel
                                                         'set ADC mode and gain
```

```
OUT adc% + 8, 1                                                    'set ADC page to 1 for digital I/O
OUT adc% + 15, &H8B                                                'setup digital port for output
DO                                                                 'wait for ADC board to be ready
x% = INP(adc% + 11)
y% = x% AND &H80
LOOP WHILE (y% <> 0)
OUT adc%, 0                                                        'Ready, so start a conversion
DO                                                                 'wait for conversion done
x% = INP(adc% + 8)
y% = x% AND &H80
LOOP WHILE (y% <> 0)
lsb% = INP(adc%)                                                   'read lower byte of conversion
msb% = INP(adc% + 1)                                               'read upper byte of conversion
dat& = lsb% + 256 * msb%                                           'pack into one 16-bit word
dat$ = HEX$(dat&)                                                  'make hex ascii string of value
DO WHILE LEN(dat$) < 4                                             'make sure conversion is 4 wide
        dat$ = "0" + dat$                                          'append a zero if too short
LOOP
frame$ = frame$ + dat$ + ","                                       'append data value to frame
NEXT K                                                             'next channel
IF LOC(2) > 0 THEN                                                 'check for commands coming in
        Cmd$ = Cmd$ + INPUT$(LOC(2), #2)                           'add command data to buffer
        1% = LEN(Cmd$)                                             'find length of command buffer
        IF 1% >= 7 THEN                                            'if there is a command there
                p% = INSTR(Cmd$, Cfs$)                             'find start of synch word
                IF p% > 0 THEN                                     'process if there is a synch word
                        cw$ = MID$(Cmd$, p% + 5, 2)                'strip out command data value
                        CE$ = MID$(Cmd$, p%, 7)                    'capture a whole command for echo
                        Cmd$ = RIGHT$(Cmd$, 1% - (p% - 1 + 7))     'trim command from buffer
                        cw% = VAL(cw$)                             'convert ascii to a number
                        IF cw% = 0 THEN                            'case 0 - reset all enable flags
                                OE% = 0
                                DSPE% = 0
                                RFE% = 0
                                RFXE% = 0
                                PT% = 0
                                CC% = CC% + 1                      'received valid command - clear flags
```

```
            EXPRUN% = &H0                          'disable all experiments
    ELSEIF cw% = 1 THEN                            'case 1 - set optics enable flag
            OE% = 1
            DSPE% = 0
            RFE% = 0
            RFXE% = 0
            CC% = CC% + 1                          'received valid command - enable Optics
            EXPRUN% = &H80                         'enable optics experiment
    ELSEIF cw% = 2 THEN                            'case 2 - set dsp enable flag
            OE% = 0
            DSPE% = 1
            RFE% = 0
            RFXE% = 0
            CC% = CC% + 1                          'received valid command - enable DSP
            EXPRUN% = &H2                          'enable DSP experiment
    ELSEIF cw% = 4 THEN                            'case 4 - set rf enable flag
            OE% = 0
            DSPE% = 0
            RFE% = 1
            RFXE% = 0
            CC% = CC% + 1                          'received valid command - enable RF
            EXPRUN% = &H20                         'enable RF experiment
    ELSEIF cw% = 8 THEN                            'case 8 - RF disabled but XMIT still on
            OE% = 0
            DSP% = 0
            RFE% = 0
            RFXE% = 0
            CC% = CC% + 1                          'received valid command - disable RF
            EXPRUN% = &H0                          'disable all experiments
    ELSEIF ((cw% = 12) AND (RFE% = 1)) THEN        'RF XMIT enable
            OE% = 0
            DSPE% = 0
            RFE% = 1
            RFXE% = 1
            CC% = CC% + 1                          'received valid command - XMIT enable
            EXPRUN% = &H28                         'enable RF XMIT experiment
    ELSEIF cw% = 16 THEN                           'case 16 - termination of sequence
```

```
                                OE% = 0
                                DSPE% = 0
                                RFE% = 0
                                RFXE% = 0
                                PT% = 1
                                CC% = CC% + 1                     'received valid command - terminate
                                EXPRUN% = &H0                     'disable all experiments
                        END IF
                END IF
        END IF
END IF
OUT adc% + 12, EXPRUN%                                            'output enable signals to relay
state% = OE% + DSPE% * 2 + RFE% * 4 + RFXE% * 8 + PT% * 16        'build state variable
state$ = HEX$(state%)                                             'convert flags to ASCII string
DO WHILE LEN(state$) < 2                                          'make sure 2-bytes wide
        state$ = "0" + state$
LOOP
'
CC$ = HEX$(CC%)
DO WHILE LEN(CC$) < 4
        CC$ = "0" + CC$
LOOP
FC$ = HEX$(FC%)
DO WHILE LEN(FC$) < 4
        FC$ = "0" + FC$
LOOP
'
frame$ = frame$ + FC$ + "," + CC$ + "," + state$ + "," + CE$      'finish data frame
PRINT #1, frame$                                                  'transmit data frame
FC% = FC% + 1
IF CC% = &H7FFF THEN CC% = 0
IF FC% = &H7FFF THEN FC% = 0
DO                                                                'wait for 1-sec timer to expire
LOOP WHILE ((TIMER - now&) < 1)
LOOP
END
```

payload.

(1) Payload Configuration

The payload configuration portion of the software is executed on payload startup. It starts at line #1 and runs through line #28. In this portion, the variables are initialized, the command and telemetry communications ports are configured, and the system is prepared for data acquisition.

(2) Telemetry Software

The telemetry processing portion of the operational software for reading the sensors is contained in lines #30 through 55. The operational flow starts by initializing the telemetry frame with the synchronization word. Then, for each data channel, the ADC board is configured for that channel and the data channel is read. Then the result of the ADC is converted to a ASCII representation of the hex values and appended to the telemetry frame.

The results of the command processing are added to the telemetry frame in lines #123 through #137. Here, the payload state variable is converted to an ASCII representation. The last valid command received is also converted to ASCII. The telemetry count, valid command count, payload state, and command echo are then appended to the telemetry frame. Finally, the telemetry frame is transmitted though the communications system. Counters and control variables are updated as needed and the software waits for the end of the timer to enable the next iteration through the processing loop.

(3) Command Software

The command processing portion of the operational software is contained in lines #56 through #122. The command processing examines the command input port for available data. If there is data available, it is read and appended to a buffer. When there are enough characters to form a valid command, the buffer is examined for a valid command word. If one is found, it is stripped from the buffer and parsed. The appropriate experiment state is set and the overall state of the payload is updated.

7. User Interface

a. Requirements

i. Each command is to be prefaced with the synchronization word "146F" prior to transmission to the payload.

ii. Command interface to prevent more than one experiment from being activated at a time.

iii. Command interface shall show the user the last command transmitted

iv. Command interface shall show the user the last command received in telemetry

v. Command interface shall require explicit user input to send a command to the payload

vi. Telemetry interface shall provide the user with the capability to see raw telemetry from payload

vii. Telemetry interface shall provide user with sensor inputs converted to measurement values with appropriate units as described in the Telemetry Table

viii. Data latency shall not exceed two seconds from the time of reception at the user interface until the data is displayed.

b. Design

The overall design of the User Interface is illustrated in Figures 18 and 19. Figure 18 shows the LabVIEW command and telemetry interface panel. Figure 19 shows the LabVIEW program diagram. Individual sections will be discussed below.

i. Command Interface

(1) User Interface

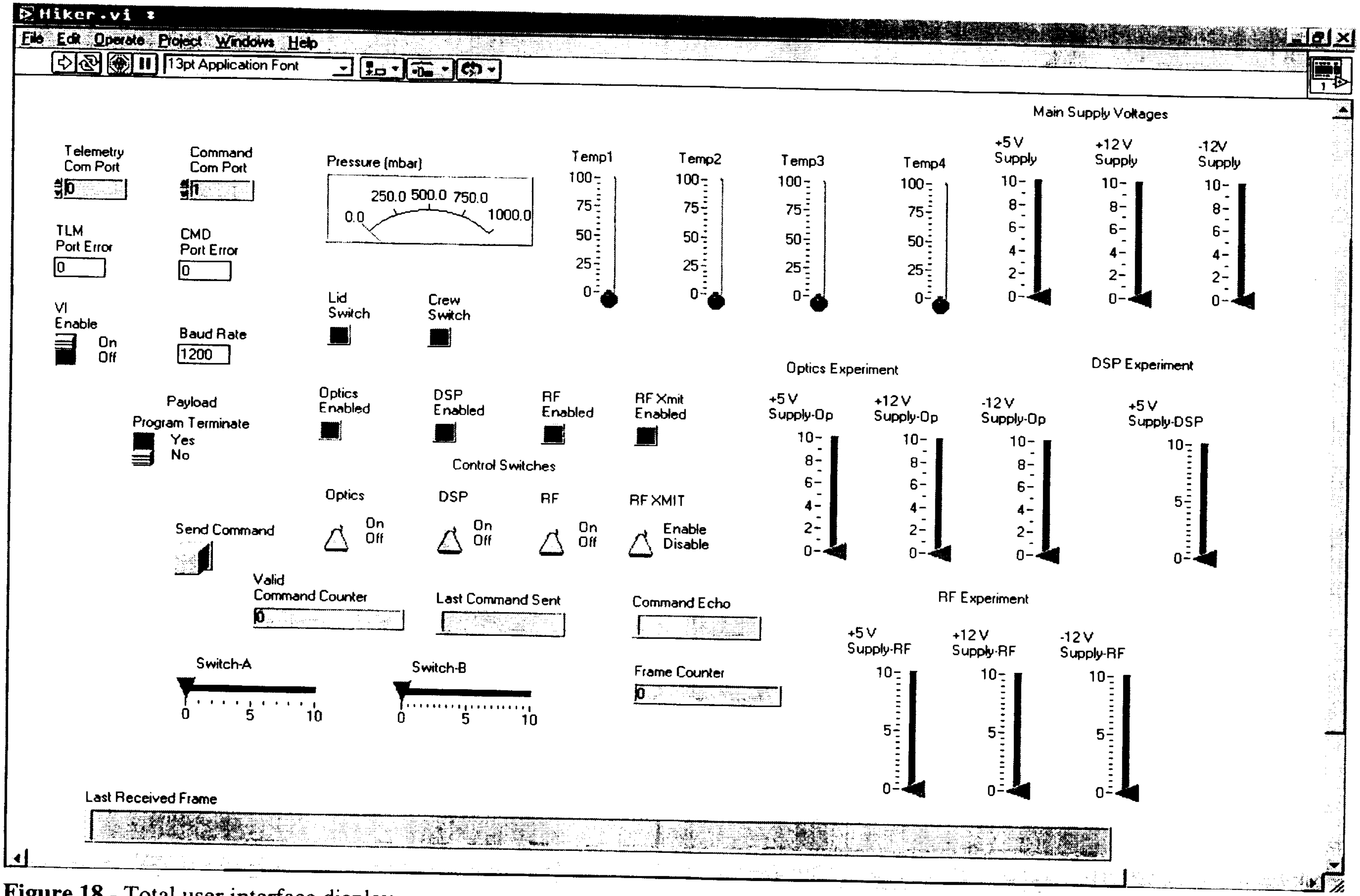
The user interface for the command processing is part of the overall payload operator user interface. The command portion is illustrated in Figure 20 with the associated program diagram in Figure 21. The user is supplied with toggle switches to enable or disable the desired experiment. To enable an experiment, the toggle switch is set to the desired state and the Send Command button is clicked with the mouse. If the command is properly received, the Enable light is illuminated over the switch. The user interface displays the last command sent, the current valid command counter from telemetry, and the command echo from telemetry.

(2) Command Processing

The command processing involves taking the current switch settings and converting them to a valid command word. The command word is then sent to the forward command link interface and displayed for the user. The LabVIEW processing performing this function is part of the overall user interface. The specific operations to perform this function is illustrated in Figure 21. The processing consists of taking the switch settings and converting them to a Boolean array. The Boolean array os then converted to a ASCII equivalent number. The command word is appended to the string "146F" which is the command synchronization word.

ii. Telemetry Interface

The telemetry interface is divided into three functional parts: the user interface, the Level 0 processing and the Level 1 processing. All telemetry interface processing was performed in LabVIEW.



Telemetry Com Port 0

Command Com Port 1

TLM Port Error 0

CMD Port Error 0

Baud Rate 1200

Valid Command Counter 0

Frame Counter 0

Figure 18 - Total user interface display.

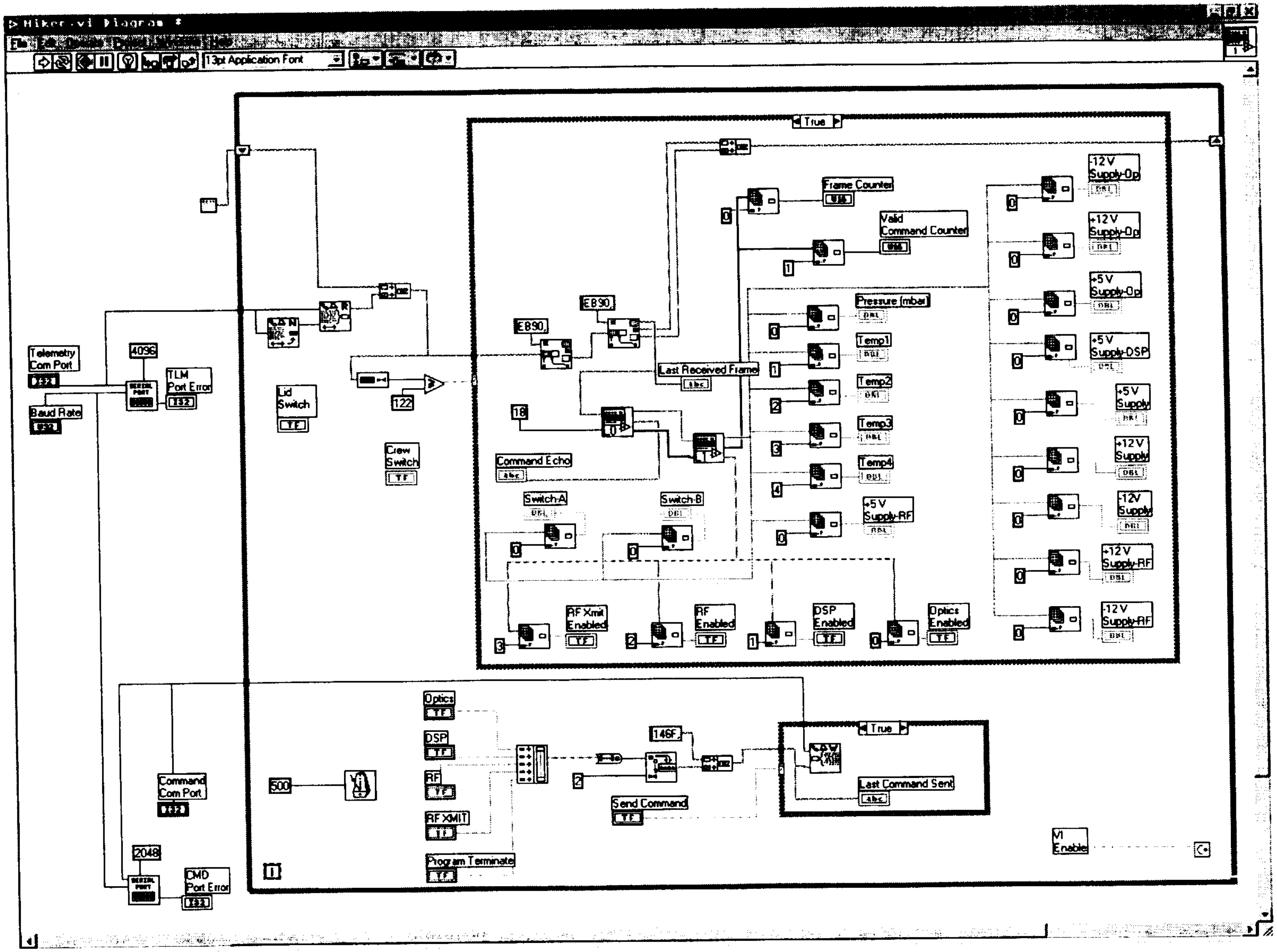


Figure 19 - User interface program diagram.

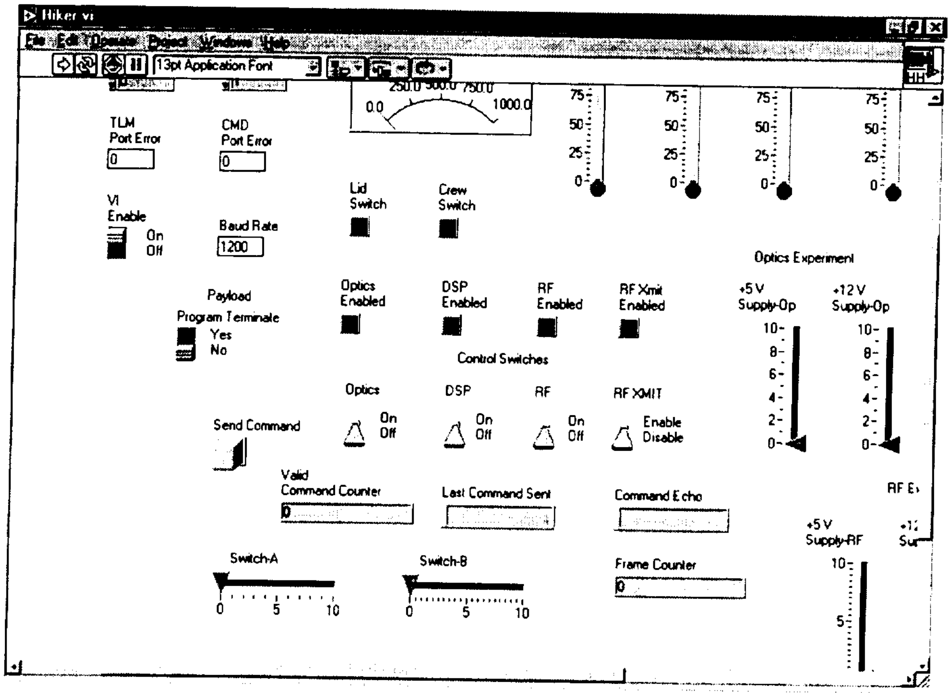


Figure 20 - Command portion of the user interface

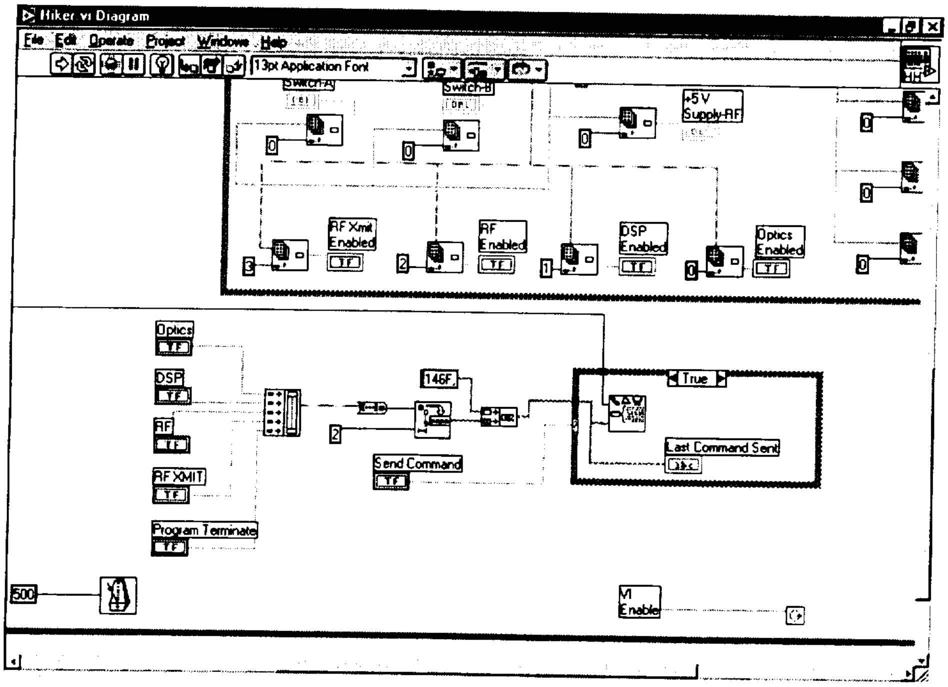


Figure 21 - Command processing portion of the overall LabVIEW user interface VI.

(1) User Interface
The user interface provides for a dial-type indicator to illustrate container pressure in millibars (mbar), four temperature thermometer-type indicators to represent temperature in degrees Celsius for the four payload locations, and the regulated +12 V, -12 V, and +5 V supply voltages, and the voltages supplied to each experiment as slide-type indicators. The user interface display is illustrated in Figure 18.

(2) Level 0 Processing
The Level 0 processing provides the initial telemetry data frame processing and sorting of data for the Level 1 processing. Level 0 processing takes the numbers from their ASCII text representation and converts them either to a number for integer-type data or to a voltage for ADC-output data. The LabVIEW interface for Level 0 processing is illustrated in Figure 22 with the associated program diagram illustrated in Figure 23.

(3) Level 1 Processing
The Level 1 processing converts voltage readings to calibrated values in units understandable to the payload operator. Level 1 processing takes the voltage values produced in Level 0 and converts them to the desired values. Level 1 processing does not change pure integer numbers or Boolean values. The pressure datum is converted to an actual value by using the manufacturer's calibration equation relating the pressure, P, in mbar to the input voltage, V:

$$\mathrm{P}(mbar) = \frac{\frac{V}{5} + 0.1}{0.0009}$$

The manufacturer's calibration equation for relating the temperature, T, in degrees Celsius to the input voltage, V, is

$$T(^{\circ}C) = 100V$$

The LabVIEW interface for Level 1 processing is illustrated in Figure 24 with the associated program diagram illustrated in Figure 25.

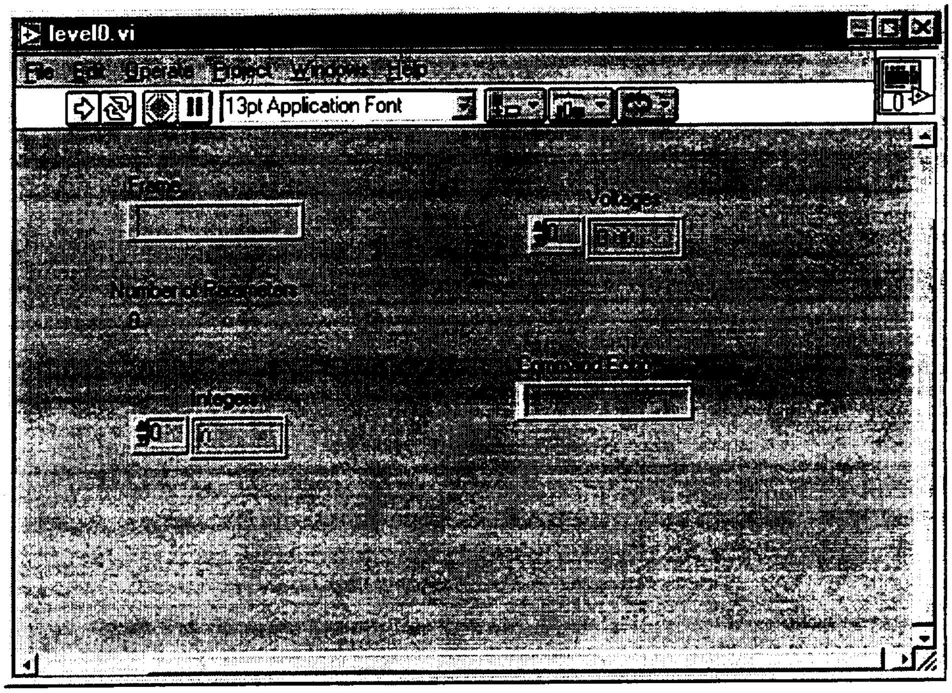


Figure 22 - Level 0 Processing LabVIEW front panel.

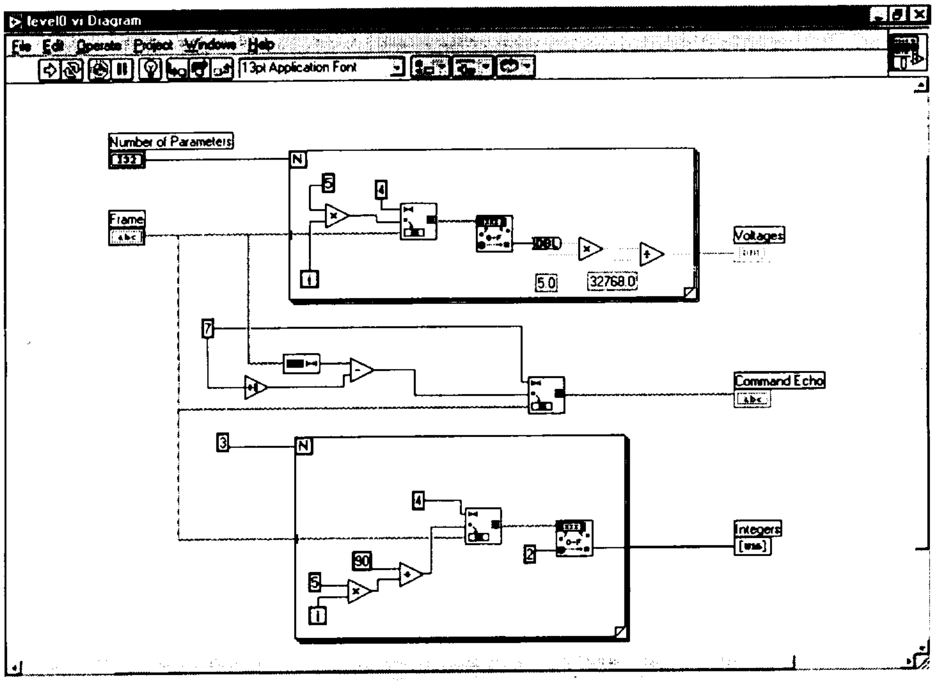


Figure 23 - Level 0 Processing LabVIEW functional diagram.

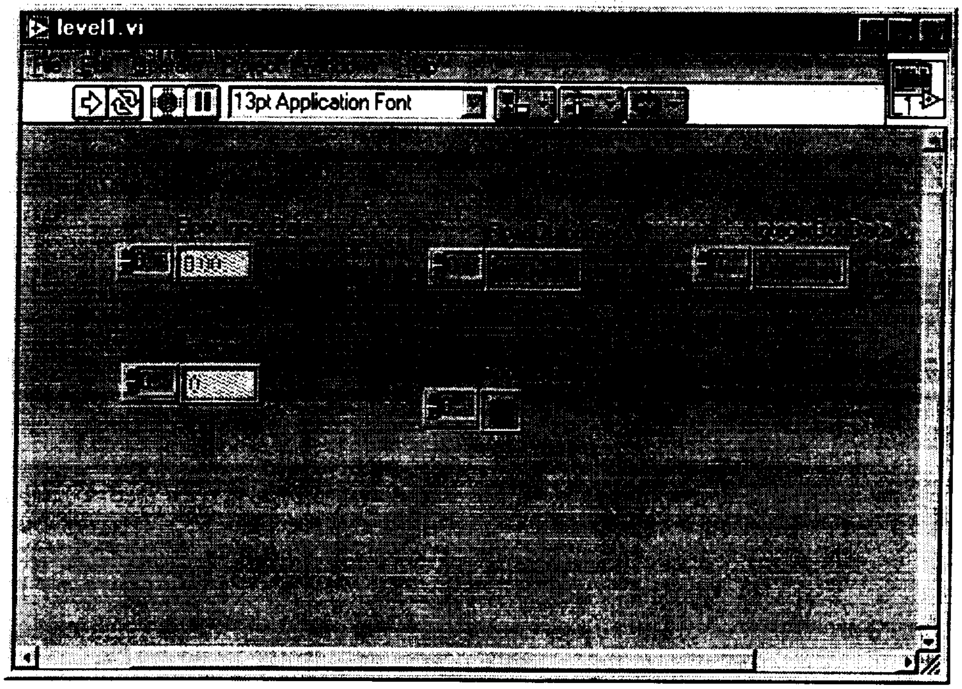


Figure 24 - Level 1 Processing LabVIEW front panel.

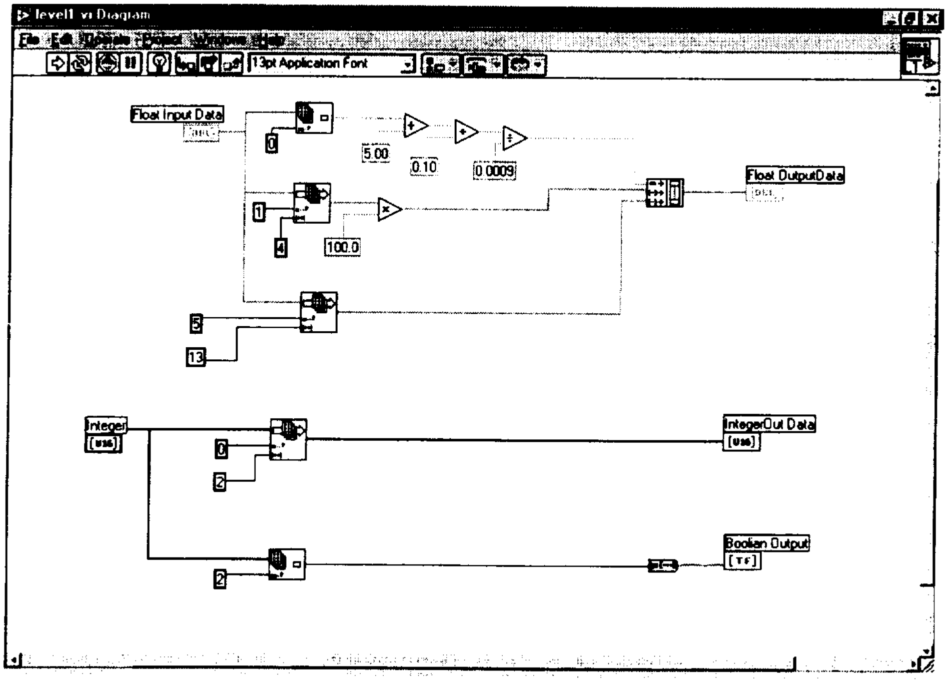


Figure 25 - Level 1 Processing LabVIEW functional diagram.

References

[1] S. Horan, “Hitchhiker Payload Mission,” NMSU-ECE-99-017, December 1999.

[2] S. Horan, “Hitchhiker System Requirements,” NMSU-ECE-99-018, December 1999.

[3] S. Horan, “NASA Hitchhiker Program Customer Payload Requirements”, NMSU-ECE-98-006, September 1998.

[4] National Aeronautics and Space Administration, “Hitchhiker Customer Accommodations & Requirements Specifications,” 740-SPEC-008, 1999.